Data Science Lab: Project Report

Andrea Cognolato
Politecnico di Torino
Student id: s281940
andrea.cognolato@studenti.polito.it

***Abstract*—This report describes the author's solution to the *Wine Quality Reviews Dataset* regression problem. The proposed approach consists of generating, using both categorical and textual data, a large ($> 65k$) number of features to be fed into a regressor. Three regression models are compared: Gradient Boosted Decision Trees, Ridge Linear Regression, and Neural Networks. An interpretability analysis is performed to explain the Ridge model's coefficients.**

I. Problem overview

The project for winter 2021's first call of the *Data Science Lab: Process and methods* exam consists of building a regression model on the *Wine Quality Reviews Dataset*. The goal of the project is to develop a model for predicting the quality score given by an expert wine taster to a specific wine.

To evaluate the performance of our models[1], in addition to a *development* dataset, the organizers have provided an unlabeled *evaluation* dataset. The latter will be used to score our predictions on an online platform[2].

The development dataset is a tabular dataset containing 9 columns and 120744 rows. The evaluation dataset contains all columns but the *quality* attribute and has 30186 rows.

Some of the dataset's categorical attributes have an high percentage of missing values, as illustrated in table I. Because of the high cardinality of some attributes we analyze how many distinct values are needed to cover 99% of the rows, in order to determine how skewed their distributions are. The results of such analysis are shown in table II. For *province*, *variety*, and *region_1* it over two thirds of the values account for less that 1% of the rows. This could be exploited in the preprocessing stage to reduce the number of attribute values and speed up training as well as potentially reduce overfitting.

TABLE I
Attributes

Attribute	Type	Number of missing values
country	Categorical	5
province	Categorical	5
variety	Categorical	0
region_1	Categorical	20008
region_2	Categorical	72008
winery	Categorical	0
designation	Categorical	36518
description	Textual	0
quality	Numerical	0

[1]Source code available at `github.com/mrandri19/polito2020-data-science-lab`
[2]Hosted at `trinidad.polito.it:8888`

> A creamed pear wine, with an attractive tang of orange zest. It is light, bright, vibrant, very fruity. The acidity does seem to be excessive, so give this wine a few months in bottle.
>
> While there is some botrytis here, this wine is not heavily into that style. Rather it shows freshness, with its white currant and grapefruit acidity and delicate perfume. There is a fine core of sweet apple skin tannins to finish.

Fig. 1. Example reviews

The *description* column contains the complete text of the expert's review, in English. Two examples reviews can be seen in figure 1.

The *quality* column, an integer between 0-100, is the quantity that needs to be predicted. Plotting its distribution, as shown in figure 2, we can observe that it is normally distributed rather than log-normally distributed, as one would expect for prices.

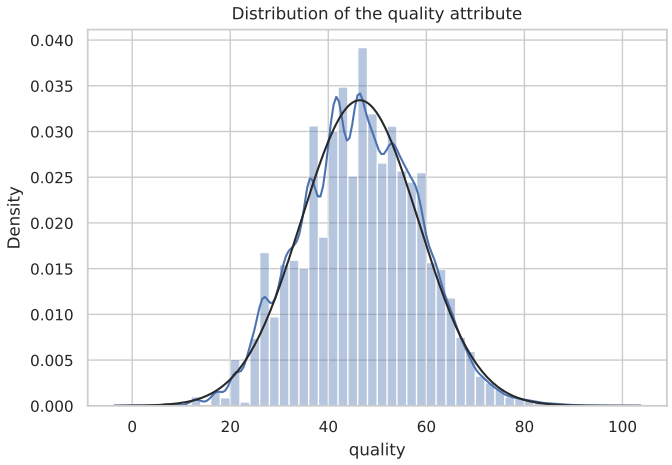

Fig. 2. Distribution of the quality attribute with Normal fit

II. Proposed approach

A. Preprocessing

The first stage in the preprocessing pipeline is a deduplication step, this alone removes over 30% of the rows in the development dataset.

TABLE II
CATEGORICAL ATTRIBUTES SKEWNESS

Attribute	p99 Count	Cardinality
country	13	48
province	186	444
variety	265	603
region_1	765	1207
region_2	15	19
winery	13253	14104
designation	26947	27798

The input data contains a mix of categorical and textual attributes. The preprocessing pipeline is therefore split into two sections, one for categorical attributes, one for textual attributes. Each section is processed separately. Finally, their outputs are concatenated and treated as input for the regression model. The pipeline was built using scikit-learn's [1] `Pipeline` and `ColumnTransformer`.

The categorical attributes have their missing values replaced by a new value, representing an unknown quantity. The actual value is not important as the next step is to One-Hot encode these attributes, thus creating a 37832-wide sparse feature matrix. Scikit-learn's `OneHotEncoder` is used to perform this transformation, with the only parameter differing from the defaults being `handle_unknown='ignore'`. This allows us to handle values never seen seen before by returning a vector of zeros. Other approaches such as Label Encoding and Target Encoding which were explored but resulted in either a decrease in performance or statistically insignificant difference. Additionally, One-Hot encoding provides interpretable features, since each attribute value maps to one and only one column, allowing us to compare them easily.

The textual attribute is vectorized using unigram TF-IDF [2], via `TfidfVectorizer`. When training on cross-validation folds, the sparse feature matrix produced is 25590-wide. Several other options for preprocessing this attribute have been explored, such as using a counting vectorizer, filtering parts-of-speech with the NLTK library [3], adding word-embeddings as features, and using both unigrams and bigrams. None of these changes resulted in statistically significant differences. Moreover, we needed a vectorizer which produced sparse, interpretable features, which restricts us from using vector-embedding based approaches.

Because both One-Hot encoding and TF-IDF return sparse matrices, their concatenation still is sparse. After concatenating them we have 63822 input features for the regressor.

B. Model selection

The following models have been tested:

- *Gradient boosted decision trees (GBDT)*: this family of models leverages an ensemble of weak learners, in this case decision trees, to build a prediction model robust against overfitting. Gradient boosting algorithms start by fitting an initial model to the data, then fitting a new model to the residuals of the old model. The new model and the old model are ensembled together obtaining a new ensemble. This procedure is repeated M times or until the error is below a threshold. We chose this model because of its good empirical performance on categorical data, observed in Kaggle competitions. We used the implementation from the LightGBM library [4].
- *Ridge linear regression*: Ridge is a linear regression model with L^2 regularization. Because the number of features is so high, almost approaching the number of samples, a regularization technique is paramount to avoid overfitting. This model was chosen initially as a baseline, because of its simplicity and interpretability. Again, we use the impementation provided by Scikit-learn.
- *Neural Network (NN)*: Because of the high number of One-Hot encoded features, as well large number of training samples, a dense feedforward neural network architecture was tested. Since neural networks are universal function approximators [5], we believe that this model could learn more complex interactions than the other two. The implementation was provided by the Keras library [6]. The architecture consisted of 3 hidden layers of widths 192, 64, 64, 1 with RELU activation functions. The output is a scalar and uses no activation function (i.e. a completely linear layer). The training process is performed using the ADAM optimizer [7].

TABLE III
GRID SEARCH PARAMETERS

Model	Parameter	Values
GBDT	num_leaves, min_child_samples	{16, 32, 64, 128},
	min_child_samples	{10, 20, 30}
Ridge	alpha	{0.01, 0.1, 1.0, 10.0}

The first two regressors have been tuned and compared using 5-fold cross validation. The Neural Network was tested using 75-25 holdout and manual hyperparameter tuning.

C. Hyperparameters tuning

The two regressors have their own set of hyperparameters listed in table III.

5-fold Cross Validation Grid Search is used to obtain the optimal hyperparameter configuration for each model, which are then compared on unseen data from the test set. The search results can be seen in figure 3.

III. RESULTS

By performing grid search on the parameter space we obtain the best hyperparameters for each model. The models are trained using the optimal hyperparameters on the whole training dataset and scored on the test dataset. The test scores are reported under the *Local* R^2 column in table IV. Finally, the three models are trained on the whole development dataset and used to label the evaluation dataset. The output is then uploaded to the submission platform, thanks to which we discover the public leaderboard score, reported in the *Public* R^2 column.

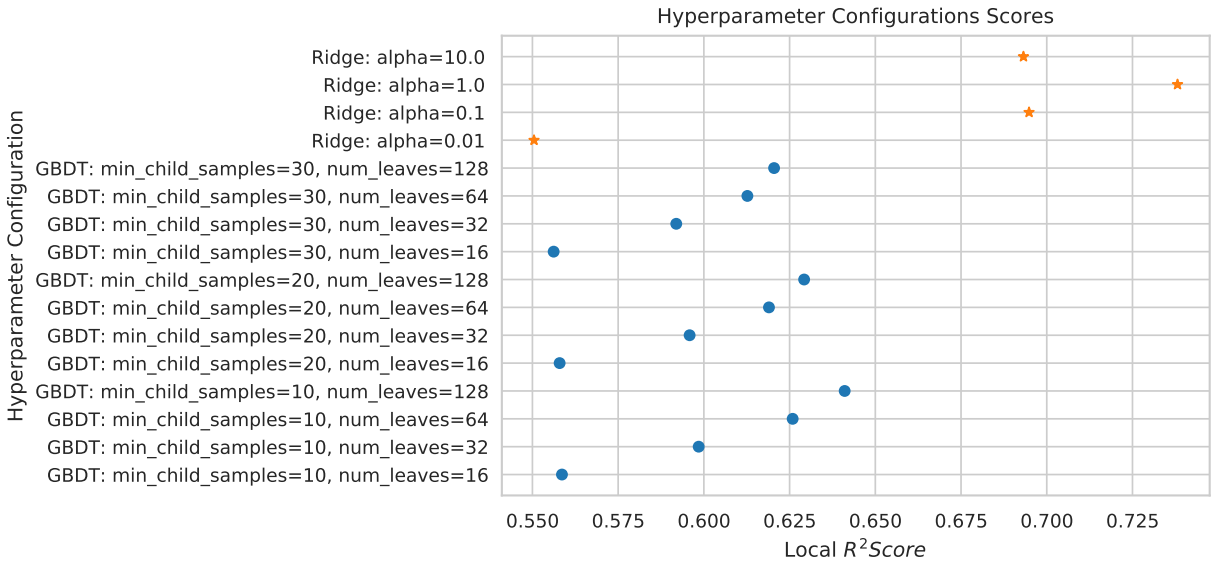

Fig. 3. Comparison of local R^2 score for all hyperparameter configurations

TABLE IV
BEST CONFIGURATIONS AND SCORES

Model	Best Configuration	Local R^2	Public R^2
GBDT	num_leaves=128, min_child_samples=10	0.639	0.709
Ridge	alpha=1.0	0.748	0.847
NN	Dense RELU (192, 64, 64, 1), Adam learning_rate=3e-3	0.752	0.851

During the development process over 20 submission to the leaderboard were made. Because of this there is a possibility that our models may not have actually improved their generalization ability, but instead overfitted the public dataset. We cannot verify this assumption without being able to look at the private score. Nonetheless we assume that this is not the case by noticing that the improvement from local to public score is constant, about 0.10. This difference remains constant even when changing the random seed used in `train_test_split`. It is unlikely that three different models, trained on different splits have overfitted the evaluation dataset by the same quantity. We can assume that the difference from local to public score is due to the entire dataset being used to train the model rather than being due to overfitting.

IV. Discussion

All three proposed approaches provide statisfatory results, beating both the provided baseline by over 0.4, and a constant baseline, whose R^2 score would be 0. The combination of textual TF-IDF feature extraction and categorical one-hot encoding provides a foundation on which many algorithms can be trained. Out of the three models considered, Ridge regression and Neural Network perform similarly, while Gradient Boosted Decision Trees achieve a lower score.

Future work to improve the results might consider the following aspects:

- More hyperparameter search, in particular for the GBDT model. A small domain of all possible hyperparameters has been explored with grid search. Running a parallel search, possibly using random search or Bayesian optimizers could result in better performance.
- New feature engineering approaches could be considered. For example by leveraging semantic word embedding to generate additional features, as proposed in [8]. Other approaches would take advantage of the automatic feature extraction provided by Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN).

To conclude, we conduct an interpetability analysis of the Ridge model in order to better understand which are the factors that contribute, both positively and negatively, to the quality of a particular wine. This also allows us to identify potential biases in the model, which could have catastrophic consequences, especially for a user-facing service.

Since Ridge is a linear model, its coefficients directly translate to an importance score for each of the features [9]. We fetch these coefficients from the trained, top-performing model found in grid search.

In table V we include the most negative features (both categorical and textual). Among this list we can notice *House Band*, which used to be plastic-bottled single-serving wine company. We can also notice *Kirkland Signature*, the discount in-house brand of Costco, one of USA's biggest warehouse store.

Table VI includes the most positive features. After further investigation we discover that these are all very selective and exclusive wineries. For instance, *Dal Forno Romano* is an historical winery, located in the province of Verona, which

TABLE V
MOST NEGATIVE FEATURES

Feature Name	Value
Napa Family Vineyards	-19.48
Bandit	-17.58
Landshut	-17.55
Terrenal	-17.19
value	-16.65
Pine & Post	-16.24
Toca Diamonte	-15.97
Two Vines	-15.78
StoneCap	-15.66
Herdade dos Machados	-15.41
Hilltown	-15.37
Oak Grove	-15.14
Funky Llama	-15.12
Backhouse	-14.93
Pavilion	-14.70
Saurus Patagonia	-14.35
Kirkland Signature	-14.32
House Band	-14.16

TABLE VI
MOST POSITIVE FEATURES

Feature Name	Value
Grange	22.97
Viñedo Chadwick	21.48
Montrachet	21.25
Kai	20.86
Rivalta Limited Selection	19.82
Blair	19.27
Gaja	18.89
Contador	18.38
40-year old tawny	18.28
Bâtard-Montrachet	17.82
Antiyal	17.21
Artadi	17.09
Chevalier-Montrachet	17.00
Finca El Bosque	16.93
Harlan Estate	16.75
Dal Forno Romano	16.73
Auma Los Lingues	16.66
Viu 1	16.55
Carmín de Peumo	16.38
La Cumbre	16.30

produces *Amarone della Valpolicella*, an highly limited and expensive bottle.

In table VII the most negative textual features are shown. The main pattern is the inexpensiveness of this bottle. Analyzing the descriptions which contain 87,88 we see that they are percentages, and that the reviewer is examining a blend of wines. The 000 is an artefact which indicates an issue in the tokenization process. By printing descriptions which contain 000 we see that it is part of numbers like *25,000*, and it's erroneously tokenized as a separate word.

Finally, in table VIII the most positive textual features are shown. Positive adjectives such as *purest* or *flagship* are present. We also see future years such as *2030, 2040*, this is because the wines under review are wines to be stored for a long time, and consumed in the future as "vintages".

TABLE VII
MOST NEGATIVE TEXTUAL FEATURES

Feature Name	Value
value	-16.65
87	-14.13
buy	-12.58
bargain	-12.47
price	-12.17
88	-11.51
inexpensive	-10.99
000	-10.39
affordable	-9.61
budget	-9.59

TABLE VIII
MOST POSITIVE TEXTUAL FEATURES

Feature Name	Value
2022	13.71
beerenauslese	12.73
2020	12.55
carmenère	12.45
eiswein	12.35
flagship	12.07
2040	11.28
2030	11.14
98	10.76
purest	10.54

REFERENCES

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. USA: Cambridge University Press, 2008.

[3] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 3146–3154, Curran Associates, Inc., 2017.

[5] B. C. Csáji *et al.*, "Approximation with artificial neural networks,"

[6] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[8] E. Lefever, I. Hendrickx, I. Croijmans, A. van den Bosch, and A. Majid, "Discovering the language of wine reviews: A text mining account," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.

[9] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.